

# Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics

Xin Li, Fuxin Li

School of Electrical Engineering and Computer Science  
Oregon State University

urumican@gmail.com, lif@eeecs.oregonstate.edu

## Abstract

Deep learning has greatly improved visual recognition in recent years. However, recent research has shown that there exist many adversarial examples that can negatively impact the performance of such an architecture. This paper focuses on detecting those adversarial examples by analyzing whether they come from the same distribution as the normal examples. Instead of directly training a deep neural network to detect adversarials, a much simpler approach is proposed based on statistics on outputs from convolutional layers. A cascade classifier is designed to efficiently detect adversarials. Furthermore, trained from one particular adversarial generating mechanism, the resulting classifier can successfully detect adversarials from a completely different mechanism as well. After detecting adversarial examples, we show that many of them can be recovered by simply performing a small average filter on the image. Those findings should provoke us to think more about the classification mechanisms in deep convolutional neural networks.

## 1. Introduction

Recent advances in deep learning have greatly improved the capability to recognize images automatically [13, 24, 8]. State-of-the-art neural networks perform better than human on difficult, large-scale image classification tasks. However, an interesting discovery has been that those networks, albeit very resistant to overfitting, would have completely failed if some of the pixels in the image are perturbed via an adversarial optimization algorithm [26, 5] (Fig. 1). One can produce images indistinguishable from the original to a human observer, but leading to significantly different results from a deep network. Those adversarial examples are dangerous if a deep network is utilized into any crucial real application, be it autonomous driving, robotics, or any automatic identification (face, iris, speech, etc.). If the result of the network can be hacked at the will of a hacker, wrong authentications

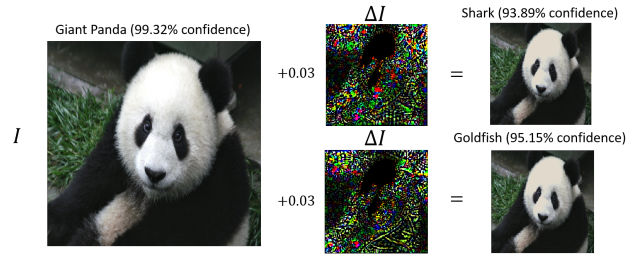


Figure 1. An optimization algorithm can find the adversarial example where, with almost negligible perturbations to human eyes, will completely distort the prediction result of a deep neural network [26]. This algorithm is quite universal, having been successfully tested on many different networks and the user can direct the network to output any category with adversarial optimization.

and other devastating effects would be unavoidable.

Therefore, there are ample reasons to believe that it is important to identify whether an example comes from a normal or an adversarial distribution. Such knowledge if available will help significantly to control behaviors of robots employing deep learning. A reliable procedure can prevent robots from behaving in undesirable manners undesirable because of the false perceptions it made about the environment.

The understanding of whether an example belong to the training distribution has deep roots in statistical machine learning. In statistical machine learning, the *i.i.d.* assumption is commonly used, so that the testing examples are assumed to be drawn independently from the same distribution of the training examples. This is because machine learning is only good at performing *interpolation*, where some training examples surround any testing example. *Extrapolation* is known to be difficult, because it is extremely difficult to estimate data labels or statistics if the data comes as extremely different from any known or learned observations. Therefore, in many cases it might be better to abstain an extrapolation prediction. Theoretical work such as [15] provides basic theoretical frameworks of classification with

an abstain option.

Although these concepts are well-known, the difficulties of understanding and applying them lie in the high-dimensional spaces that are routinely used in machine learning and especially deep learning. Is it even possible to define interpolation vs. extrapolation in a 4,000-dimensional or 40,000-dimensional space? It looks like almost everything is extrapolation since the data is inherently sparse in such a high-dimensional space [9, 7], a phenomenon well-known as the curse of dimensionality. The enforcement of the *i.i.d.* assumption seems impossible in such a high-dimensional space, because the inverse problem of estimating the joint distribution requires an exponential number of examples to be solved efficiently. Some recent work on generative adversarial networks propose using a deep network to train this discriminative classifier [4, 21], but that requires an abundant number of examples and a significant amount of parameters. A generative approach is required to generate those samples, but largely was confined to unsupervised settings and may not be applicable for every domain convolutional networks have been applied to.

In this work we propose a discriminative approach to identify adversarial examples, which trains on simple features and can approach good accuracy with few training examples. Our idea comes from intuitions that show that deep learning finds manifolds of much lower dimensionality and utilizes those low-dimensional manifolds to build efficient learning machines. We make a number of empirical visualizations that show how the adversarial examples change the prediction of a deep network. From those intuitions, we extract simple statistics from convolutional filter outputs from various layers in the CNN. A cascade classifier is proposed that utilizes features from many layers to discriminate between normal and adversarial examples.

Experiments show that our features from convolutional filter output statistics can separate between normal and adversarial examples very well. Trained with one particular adversarial generation method and a couple thousand examples, it is robust enough to generalize adversarials produced from another adversarial generation approach without any special adaptation. Those confidence estimates may improve the safety of applying these deep networks, and hopefully provide insights for further research on self-aware learning. As a simple extension, the result from visualizations of the features prompted us to perform an average filter on corrupted images, and found out that many correct predictions can be recovered from this simple filtering.

## 2. Deep Convolutional Neural Networks

A deep convolutional neural network (CNN) consists of many different layers (Fig. 2), where each layer is connected to spatially/temporally adjacent nodes to the next

layer:

$$\mathbf{Z}_{m+1} = [T(\mathbf{W}_1 * \mathbf{Z}_m), T(\mathbf{W}_2 * \mathbf{Z}_m), \dots, T(\mathbf{W}_k * \mathbf{Z}_m)] \quad (1)$$

where  $\mathbf{Z}_m$  is the input features at layer  $m$ ,  $\mathbf{W}_1, \dots, \mathbf{W}_K$  are filters that could be much smaller than the size of  $\mathbf{Z}_m$  (e.g.  $3 \times 3, 5 \times 5, 7 \times 7$ ),  $*$  is the convolution operator, and  $T$  is a nonlinear transformation function such as the rectified linear unit (ReLU)  $T(x) = \max(0, x)$ . This network would apply the same convolutional filters at every location of  $\mathbf{Z}_m$ , which means that the number of parameters is much smaller than the size of  $\mathbf{Z}_m$ . Other commonly used layers in a CNN include max-pooling layers, or other normalization layers [13] such as batch normalization layers [10]. Max-pooling selects the maximal output from a small region, in order to allow minor deformations for prominent features detected from the convolutional filter. Most deep networks adopt similar principles while adding more structural complexity in the system such as more layers and smaller filters in each layer [24], multi-layered network within each layer [25], combining multiple networks [29], etc. A convolutional neural network makes sense in structured data because it naturally exploits the locality structure in data. In an image, pixels that are located close to each other are naturally more correlated than pixels that are far away. The same holds for temporal data (video, speech) where objects (frames, utterances) that are temporally close can be assumed to be more correlated.

The 16-layer VGG network in Fig. 2 serves as a good example. This network contains 14 convolution layers, each featuring an increasing number of  $3 \times 3$  filters (2). It also has 5 max-pooling layers in the middle, and at the end 2 fully connected layers before the classification layer. The task that it performs is ImageNet classification [3], to classify natural images in a photo collection into 1,000 different categories. It was trained on a training set of 1.2 million images and has a testing top-5 error rate of 13.5%.

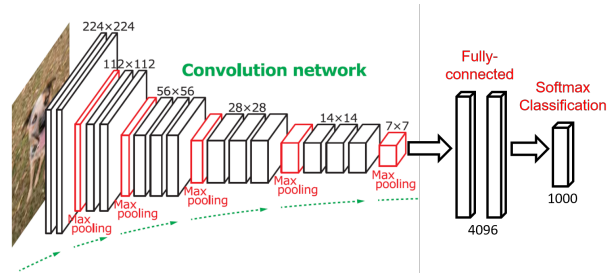


Figure 2. (a) A sample configuration of a state-of-the-art deep convolutional network architecture ([24], figure modified from [19]). The network consists of convolutional layers, max pooling layers (maxpool) [13] and fully connected layers (where nodes are connected to every node from the previous layer). Finally, a soft-max layer computes a logistic loss for classification. ReLU nonlinearity is used for every layer.

### 3. Understanding the Trained Deep Classifier Under Adversarial Optimization

#### 3.1. Adversarial Optimization

The famous result that deep networks can be broken easily [26] is an important motivation of this work. The idea is to start from an existing example (image) and optimize to obtain an example that will be classified to another category while being close to the original example. Namely, the following optimization problem is solved:

$$\begin{aligned} \min_r \quad & c\|r\|_1 + L(f_\theta(\mathbf{x}_0 + r, y)) \\ \text{s.t.} \quad & \mathbf{x}_0 + r \in [0, 1]^d \end{aligned} \quad (2)$$

where  $\mathbf{x}_0$  is a known example and  $y$  is an arbitrary category label,  $d$  is the input dimensionality.  $c$  is a parameter that can be tuned for trading off between proximity to the original example  $\mathbf{x}_0$  and the classification loss on the other category  $y$ . It has been shown, to the astound of many, that one can choose an  $r$  with very small norm while completely change the output of the algorithm (e.g. Fig. 1), this can be even done universally for almost all networks, datasets and categories [26, 5]. This has led many people to question whether deep networks are really learning the “proper” rules for classifying those images.

#### 3.2. Adversarial Behavior

In order to gain a deeper understanding of the behavior of a deep network, we utilize spectral analysis. As a starting point, we perform principal component analysis (PCA) [11] at the 14-th layer of the network (the first fully-connected layer). The rationale behind using PCA is that each deep learning layer is a nonlinear activation function on a linear transformation, hence a lot of the learning process lies within the linear transformation, for which PCA is a standard tool to analyze.

A linear PCA is performed on the entire collection of 50,000 images from the ImageNet validation set, as well as 4,000 adversarials collected using the approach in (2), starting from random images in the collection. The result shows very interesting findings (Fig. 3) and shed more light on the internal mechanics of those adversarial examples. In Fig. 3(a), we show the PCA projection onto the first two eigenvectors. This cannot separate normal and adversarial examples, as one could possibly imagine. The adversarial examples seem to exactly belong to the same distribution as normal ones. However, it does seem that the adversarial examples reside mostly in the center while the normal examples occupy a bigger chunk of space.

Interestingly, as we move to the tail of the PCA projection space, the picture start to change significantly. In Fig. 3(b), we can see that there are a significant amount of adversarial examples that has extremely large values w.r.t.

to the normal examples in the tail of the distribution. We chose to print the projection on the 3,547-th and 3,844-th eigenvector, but similar distributions can be found all over the tail. As one can see, at such a far end on the tail, the projections of normal examples are very similar to random samples under a Gaussian distribution. An explanation for that could be that under these “uninformative” directions, most of the weighted features are nearly independent w.r.t. each other, hence the distribution of their sum is similar to Gaussian, according to the central limit theorem<sup>1</sup>. However, although normal examples behave similarly to a Gaussian, some adversarial examples are having projections with a deviation as large as 5 or 10 times the standard deviation, which are extremely unlikely to occur under a Gaussian distribution.

Fig. 3(c) and Fig. 3(d) shows that there are two distinct phenomena:

- The extremal values and standard deviations on the projections onto the first 500 – 700 eigenvectors are decidedly lower in adversarial examples than in normal ones.
- The extremal values and standard deviations on the projections onto the last 1,000 – 1,500 eigenvectors are decidedly higher in the adversarial examples than the normal ones.

It is interesting to reflect about the causes and consequences of those properties. One deciding property is that there is a strong regularization effect in adversarial examples on almost all the informative directions. Hence, the predictions in adversarial examples are *lower* than those of normal examples, rather than the confidence values may have indicated (Fig. 1). In Fig. 4, we show the number of categories with a prediction higher than a threshold, before the final softmax transformation

$$p_i(x) = \frac{\exp(f_i(x))}{\sum_i \exp(f_i(x))} \quad (3)$$

that converts raw predictions  $f_i(x)$  into probabilities. The result shows that normal examples have on average one category with a raw prediction value more than 20, however adversarial examples have only 0.01 category with raw predictions more than 20. The reason that those adversarial examples appear more *confident* after softmax is because that the predictions on all the other categories are regularized *even more*. Hence the normalization component of softmax has decided that the single prediction, although much less strong, should be assigned a probability of more than 90%. We note that this issue was also pointed out by [2] and they proposed a solution in the OpenMax classifier.

<sup>1</sup>Note this is directly after the final convolution layer without a ReLU transformation, since ReLU destroys the negative part of the data distribution, the data no longer looks like a Gaussian after ReLU. However, some tail effects can be observed even in the distribution after ReLU.

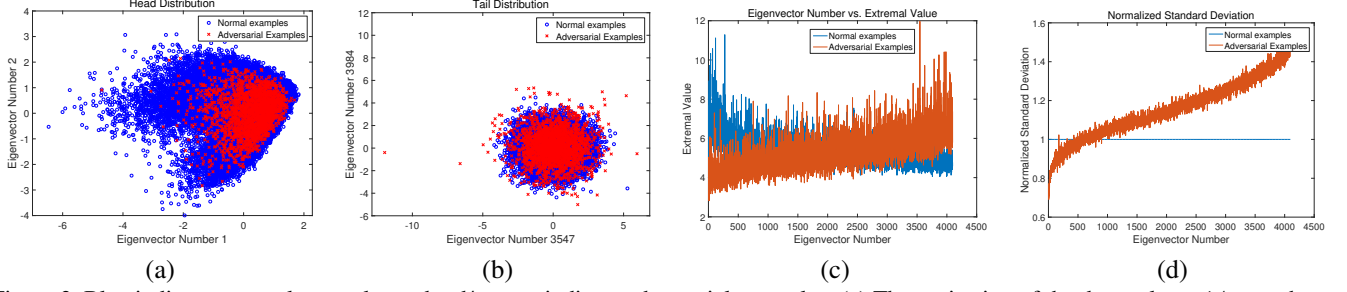


Figure 3. Blue indicates normal examples and red/orange indicate adversarial examples. (a) The projection of the data at layer 14 onto the 2 most prominent directions; Adversarial example cannot be identified from normal ones. (b) Projection of the same data to the 3,547-th and 3,844-th PCA projections, some adversarial examples are having significantly higher deviation to the mean; (c) The absolute value of the most extremal value in the projection to each eigenvector, normalized by the standard deviation of the projections on the normal eigenvectors; (d) The average normalized standard deviation of normal and adversarial examples on each projection.

But besides that, it seems that such extremal and standard deviation features are evident features that could help discriminating normal and adversarial examples. Unfortunately, they only occur as a statistic from a large sample, as any single point in Fig. 3(a) looks similar to a single point in the normal distribution. We have tried to utilize the tail distributions (Fig. 3(b)) to create a classifier, however we subsequently found out that those tail distributions almost do not contribute to the classification, and the adversarial example can easily optimize to remove their footprints on the tail distributions.

This leads us to think about an approach that would turn a single image into a distribution. Arguably, an image is a distribution of pixels, especially in the convolutional layers. The output of each filter from each convolutional layer is an image which could be treated as a distribution where the samples are the pixels. Therefore, in the following section we aim to build a classifier based on collecting statistics from such distributions.

## 4. Identifying Adversarial Examples

### 4.1. Feature Collection

Suppose the output at a convolutional layer  $m$  is an  $W \times H \times K$  tensor, where  $W$  and  $H$  represent the width and height of the image at that stage (smaller than original after max-pooling), and  $K$  represents the number of convolutional filters. Such a tensor can be considered as a  $K$ -channel image where each pixel has a  $K$ -dimensional feature. We consider the feature on every pixel to be a random vector drawn from the distribution  $D_m$  of convolutional pixel outputs, a  $K$ -dimensional distribution.

The list of statistics we collect is:

- Normalized PCA coefficients
- Minimal and Maximal values
- 25-th, 50-th and 75-th percentile values

on each of the  $K$ -dimensional outputs. Normalized PCA coefficients are collected via Algorithm 1. Extremal and percentile statistics are straightforward to understand.

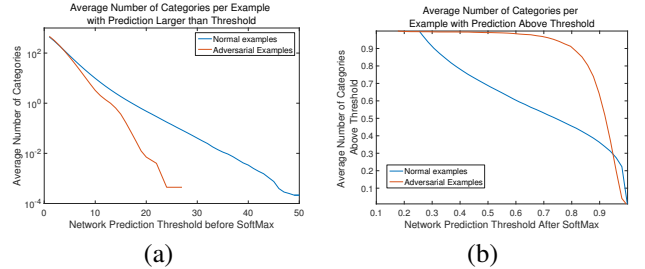


Figure 4. Average number of categories per example with predictions higher than a threshold. (a) Before softmax; (b) After softmax. As one can see, in normal examples, there are on average about 1 category with a prediction score of more than 20 (before softmax), while with adversarial ones, only 1% examples have a category with a prediction score more than 20. However, since prediction values on all categories have dropped, after softmax adversarial examples obtain much higher likelihood on one category.

---

#### Algorithm 1 PCA Statistics Extraction

---

- 1: Draw examples from the  $D_m$  of normal images in a training set to form an example matrix  $\mathbf{Z}$ .
  - 2: Compute the mean  $\mathbf{m}$  and PCA projection matrix  $\mathbf{W}$  of  $\mathbf{Z}$ .
  - 3: Compute the standard deviation  $\mathbf{s}$  on each dimension in the PCA projection  $\mathbf{W}^\top(\mathbf{Z} - \mathbf{m}\mathbf{1}^\top)$ .
  - 4: For each image  $I$ , draw  $n$  pixels from this image  $\mathbf{Z}_I$ , project them using PCA:  $\mathbf{W}^\top(\mathbf{Z}_I - \mathbf{m}\mathbf{1}^\top)$ , and normalize them by dividing the standard deviation on each respective dimension.
  - 5: Collect the statistic for each image as  $\mathbf{x}_I = \frac{1}{n} \|\mathbf{Z}_I\|_1$ , where  $L_1$  norm is the vector  $L_1$  norm on each PCA projection dimension, thus the resulting statistic is  $K$ -dimensional.
- 

### 4.2. Classifier Cascade

[27] proposed a famous strategy to detect patches of pixels that represent human faces, a cascaded boosting classifier composed by a sequence of base classifiers. A cascade classifier is ideal when it is easy to identify many of the

examples from a category but some important cases can be difficult. In Fig. 5,  $SC\_N$  represents the classifier at each stage.  $X$  is the input of the cascade classifier. The negatives in a cascade classifier from each stage will be outputted directly, while the positive will go to the next stage.

In our case, the normal category is much easier to detect than the adversarial category (see e.g. Fig. 7). In our initial experiments with VGGNet, we found that more than 80% of normal examples can be determined from the first convolutional layer with 0% false positives. Therefore, we constructed a cascade classifier based on convolutional layers: the first stage works with features collected from the outputs of the first convolutional layer, the second with the second layer, etc.(Fig. 5). The base classifiers will not solely consider statistics from their own stage, instead, after one stage of training, the remaining positive examples will be concatenated to the corresponding features on the next stage. The training task is harder at the latter stages of the cascade.

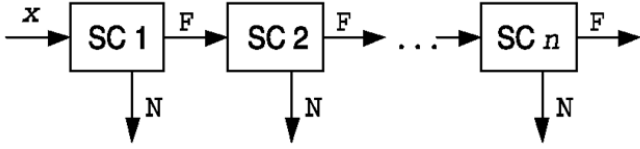


Figure 5. RCO for each of the convolutional layers in AlexNet

The operations that represented by Fig.5 can also be summarized as Algorithm.2.

---

**Algorithm 2** Training Process of a cascade of Classifier

---

- 1:  $N_{pool} \leftarrow$  Normal example pool,  $P_{train} \leftarrow$  Training set of  $N_p$  perturbed examples,  $L \leftarrow$  Number Of convolutional layers
  - 2: **while** current layer  $\leq L$  **Or**  $N_{pool} \neq \emptyset$  **do**
  - 3:   Draw  $N_p$  sized subset  $P_{normal}$  from  $N_{pool}$
  - 4:    $T \leftarrow P_{normal} \cup P_{train}$
  - 5:   Train SVM on  $T$
  - 6:   Predict SVM on  $N_{pool}$ , eliminate those predicted as normal above a threshold (described in text)
  - 7: **end while**
- 

The overall false positive rate of a  $K$  stage cascade classifier can be represented as:  $F = \prod_{i=1}^K f_i$ , where  $f_i$  is the false positive rate at each layer. And similarly the true positive rate can be represented in the same form:  $T = \prod_{i=1}^K t_i$  where  $t_i$  is the true positive rate at each stage. In order to maximize recall, we maintain a high true positive rate and select a classification threshold which corresponds to a relatively high true positive rate (97% in AlexNet and 98% in VGG).

## 5. Related Work

Szegedy et al. [26] proposes the adversarial optimization formulation in (2). [5] proposes a nice explanation of the adversarial mechanism, in which it pointed out that the linearity of the network was the cause that the network is easily corruptable, and proposed a simpler adversarial optimization mechanism that only corrupts based on the signs of gradient of the network. The fact that such examples can be generated so easily with the gradient sign method shows that adversarial examples come from attacking the magnifying effect coming from the linearities in the network. [18] proposes another mechanism to generate adversarials using evolutionary optimization. The result of these do not resemble natural images but still can be classified by deep networks with high confidence(Fig. 6). [17] proposes another efficient approach. [22] proposes an approach to generate adversarials that match the convolutional filter outputs as well as perturbing the data.

Recently, there have been a lot of focus on training adversarial generation networks to create Generative Adversarial Networks (GANs) [4, 21, 30, 23]. These networks play a two-player game where a generator network aims to generate adversarials that will not be correctly classified by another discriminator network, and the goal is to generate images more and more similar to natural images. It has been shown that these networks generate images that resemble natural images. However, this generative approach is different from our goal, where we aim to create discriminative networks that discriminates from images that are already indistinguishable from natural images (e.g. Fig.1).

Mechanisms for detecting and countering adversarial examples have also been proposed [6, 20]. [16] proposes to use the foveation mechanism to alleviate adversarial examples when it is already known to be adversarial, but did not attempt to detect adversarials. The open-set deep networks proposed by [2] seek to alleviate concerns from a soft-max classification by creating an abstain option.

Self-aware learning (classification with an abstain option) had been proposed in e.g. [12, 15]. It is relevant to robust learning (e.g. [14]), however robust learning usually seek to directly optimize the minimax loss under adversarial conditions, instead of outputting an abstain option. [1, 28] also focuses on classification with an abstain option.

## 6. Experiments

Our algorithm is tested on 2 data sources. The main one is data generated using the L-BFGS algorithm by [26]. We generated 4,000 adversarials from a random subset of the ILSVRC-2012 validation set (total of 50,000 images). In order to test the out-of-sample generalization capability, we included another dataset, which includes 5,000 EA-adversarial images generated using the algorithm in [18].



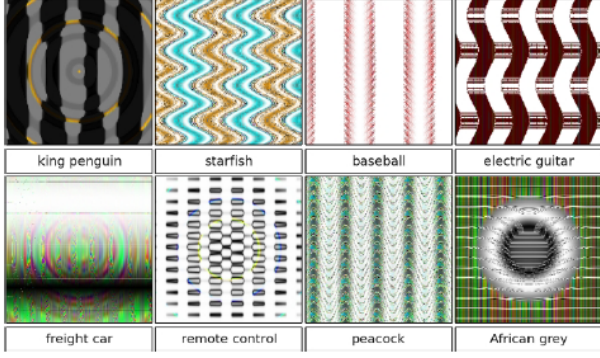


Figure 6. Evolved Adversarial Images Unrecognizable to Humans

These datasets are tested with 2 different network structures, VGG-16 [24] and AlexNet [13]. All input images had been normalized and re-shaped into  $224 \times 224 \times 3$  dimensions for VGG-16 model and re-shaped into  $227 \times 227 \times 3$  dimensions for AlexNet. The MatConvNet toolbox from VLFeat is used for the deep networks. **LinearSVC** from the **scikit-learn** toolkit in Python 2.7 is used for the SVM classifier. Features are extracted from the statistics of interest, mentioned in section 4.1, in the lower 9 convolutional layers from the VGG-16 network, and in all 5 convolutional layers from AlexNet.

All the classifiers were trained with a subset of 2,000 L-BFGS adversarials and a randomly selected subset of 45,000 normal images in the ILSVRC-2012 dataset. Testing were done on a randomly selected held-out dataset with the other 2,000 normal images and 2,000 L-BFGS adversarials for the L-BFGS experiments, and on a randomly selected 5,000 normal images and 5,000 EA-adversarials for the EA experiments. EA experiments are only conducted with AlexNet since we found out that EA-adversarials did not succeed fooling the VGG-16 network. We have also tried some other fast adversarial generation approaches such as [5], but they generally do not generate adversarials that fool classifiers as completely (with more than 90% confidence) or as universally (on 100% of the images) as the L-BFGS algorithm.

### 6.1. Single-Layer Results

The first experiments we did were by extracting statistics on each single convolutional layer output, instead of using the cascade classifier. From Tables 1 and 2, one can see that the statistics from a single layer has some discriminative power to distinguish normal examples from adversarials, but are not extremely effective. However, EA-adversarials are much easier to distinguish, even if the classifier is trained on L-BFGS instead of EA (Table 3). We only need the data generated from the first three convolutional layers to reach overall 97.34% classification accu-

racy. We believe the reason is that the features capture natural image statistics, and because EA-adversarials look so unnatural, their statistics are vastly different than natural images (see Sec. 6.3 for more discussions).

Table 1. Classification Result with AlexNet for Normal vs. LBFGS-adversarials

Network Layer	2nd	3rd	4th
Accuracy	$57.5 \pm 0.7$	$67.3 \pm 0.7$	$70.9 \pm 0.6$
Network Layer	5th	6th	
Accuracy	$74.9 \pm 0.9$	$78.95 \pm 0.6$	

Table 2. Classification Result with VGG-16 for Normal vs. LBFGS-Adversarials

Network Layer	2nd	3rd	4th
Accuracy	$72.1 \pm 0.7$	$84.1 \pm 0.7$	$80.3 \pm 0.6$
Network Layer	5th	6th	7th
Accuracy	$81.4 \pm 0.9$	$74.3 \pm 0.6$	$73.9 \pm 0.6$
Network Layer	8th	9th	10th
Accuracy	$74.2 \pm 0.7$	$71.2 \pm 0.7$	$74.3 \pm 0.8$

Table 3. Classification Result for Normal vs. EA-Adversarials

Layer	2nd	3rd	4th
Accuracy	$93.45 \pm 0.69$	$98.3 \pm 0.73$	$97.9 \pm 0.57$

### 6.2. Experiment for LBFGS-Adversarials Detection

Next we test the cascade classifier with both AlexNet and VGG-16. The parameter  $C$  is set to 0.005. The average accuracy of the cascade classifier reaches 83.4% over 20 random trials, and the AUC (area-under-curve) metric is 90.7%. We compared against the recently published OpenMax method [2]. To learn the Weibull distribution required for OpenMax, the EVT is applied on the same training set as the algorithm. Figure 7(a) shows the comparison results, where we were able to outperform OpenMax by over 9% in AUC (area-under-curve) measure and 11% in terms of accuracy.

In VGG-16, the results are even better. The accuracy of the classifier was on average 90.665% over 20 random trials. Fig.7(b) shows the ROC curve. We believe the fact that VGG has a lot more layers than AlexNet helps setting more constraints on the layer statistics, and is subsequently helpful for detecting adversarial examples. While we have not tested on it, we expect the results on deeper networks such as ResNet [8] to be even better.

Finally, the cascade classifier was tested on EA-adversarials. We obtained more than 96% accuracy with 0 false positive rate, with a final accuracy of 97.3% and AUC of 98.2% (Fig.7(c)). In other words, our algorithm is rarely fooled by EA-adversarials.

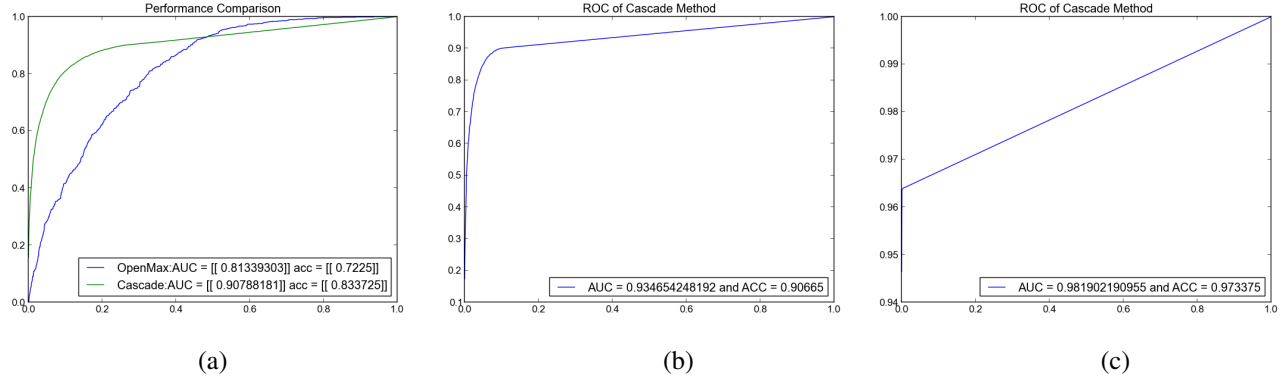


Figure 7. (a) Comparison Between OpenMax detection Methods and Cascade Classifier: The blue curve represents the performance of OpenMax Method, and green curve represents the performance for Cascade Classifier. (b) Overall ROC Performance Curve of Cascade Classifier Trained on VGG-16 Network. (c) Overall ROC of data generated from EA-adversarials dataset on AlexNet.

### 6.3. Visualization of Statistics

Our experiment results show that EA-adversarials are easy to detect with our detector. To gain more insight into this result, we made a few comparisons between the statistics of interest extracted from normal images, LBFGS-adversarials and EA-adversarials.

We visualized the average of the statistics that are used for the detection task from the first layer of the AlexNet on all its dimensions. As we can see in Fig. 8(a), the difference on the PCA projection statistics on extracted from EA-adversarials and that of the normal images is very dramatic. Meanwhile, compared to the EA-adversarials, the statistics from LBFGS-adversarial have much less difference from the normal data and the difference does not change very much across different dimensions.

From Fig. 8(b), one can see that LBFGS-adversarials have smaller extremal values than normal images. This might imply that the LBFGS optimization worked to diminish strong signals from the original image by introducing small pixel perturbations, and that helped our classifiers separating them from normal images. From Fig. 8(c), we still see the EA-adversarials evidently differ from normal images. Those results illustrate why EA-adversarials are easier to detect. We suspect it would be easy to reach 100% accuracy, if we had actually trained on some EA-adversarials. However, the capability to generalize to EA-adversarials from training on LBFGS-adversarials showed the general capability of our cascade classifiers to capture natural image statistics and distinguish natural images from unnatural ones.

## 7. Discussions

### 7.1. Self-Aware Learning with an Abstain Option

The framework of self-aware learning [15] considers the case where the learning algorithm has an abstain option of saying “I don’t know”, instead of making actual prediction

on the example. We define a framework that is slightly different than [15], avoiding the KWIK requirement of never making a mistake.

We assume that the training input is drawn i.i.d. from a distribution  $P(\mathbf{x}, y)$ , where  $\mathbf{x}$  is the input and  $y$  is the output. Assume that the testing input is drawn from a mixture distribution between  $P(\mathbf{x}, y)$  and  $Q(\mathbf{x}, y)$ :

$$P_m = \Omega P(\mathbf{x}, y) + (1 - \Omega)Q(\mathbf{x}, y) \quad (4)$$

, where  $\Omega \in \{0, 1\}$  is an unknown mixture weight, and  $Q(\mathbf{x}, y)$  is an adversarial distribution. Assume that we have a classifier that includes a function  $f(\mathbf{x})$ , and a boolean strategy  $a_i$  between `predict` and `abstain` that can be chosen for each individual  $\mathbf{x}_i$ . Assume that the expected error from our classifier on the adversarial distribution is  $e_q$  (which could be assumed, if no other prior is present, as the random guessing error of  $\frac{C-1}{C}$  for a  $C$ -class classification problem). Further assume that abstaining always incur a fixed cost  $e_a$ . As long as  $e_a < e_q$ , abstaining would be better than predicting on the example drawn from the adversarial distribution, however,  $e_a$  should be set sufficiently large so that it still makes sense to predict if the classifier is confident about it.

For each testing input, the testing of the self-aware classifier is then trying to optimize  $\min_a E_{P_m} L_a(\mathbf{x}, y)$  where

$$L_a(\mathbf{x}_i, y_i) = \begin{cases} P(y_i \neq f(\mathbf{x}_i)), & \text{if } a_i = \text{predict}, \\ & (\mathbf{x}_i, y_i) \sim P(\mathbf{x}, y) \\ e_q & \text{if } a_i = \text{predict} \\ & , (\mathbf{x}_i, y_i) \sim Q(\mathbf{x}, y) \\ e_a & \text{if } a_i = \text{abstain} \end{cases} \quad (5)$$

hence the classifier needs to select between making a prediction using its function  $f(\mathbf{x})$  and risk paying  $e_q$  versus abstaining. It is then not difficult to see that the optimal

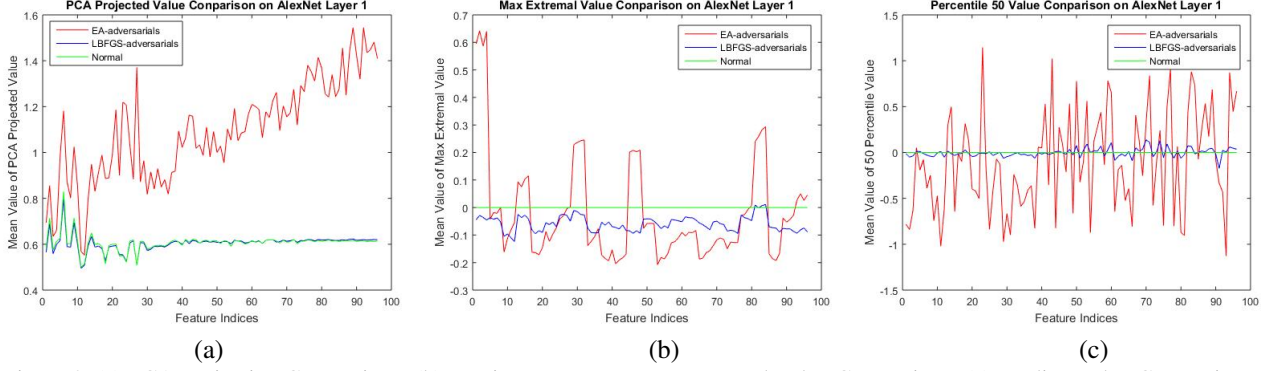


Figure 8. (a) PCA Projection Comparison; (b) Maximum Feature Map Extremal Value Comparison; (c) Median Value Comparison

strategy is:

$$a_i = \text{predict}, \quad \text{if } P(\Omega = 1|x_i)P(y_i \neq f(x_i)) + P(\Omega = 0|x_i)e_q < e_a \quad (6)$$

$$a_i = \text{abstain}, \quad \text{otherwise} \quad (7)$$

which leaves the question of estimating  $P(\Omega = 1|x_i)$ , whether the image  $x_i$  comes from the training distribution or the adversarial distribution. Our approach can be seen as estimating  $P(\Omega = 1|x_i)$  in this framework and we eagerly hope to apply it in realistic applications in future work.

## 7.2. Image Recovery

Insights from [5] indicates that the adversarial mechanism is very specifically attacking vulnerable gradients starting from the first convolutional layer. Insights from the previous experiment also suggests that LBFGS-adversarials work to diminish filter responses from the first convolutional layer. Therefore a natural idea would be to destroy the adversarial effects in the first convolutional layer to try to recover the original image. We try a very simple approach: applying an small (e.g.  $3 \times 3$ ) average filter on the adversarial image before using CNN to classify it. The positive and negative adverse gradients will average out in this approach, and making the masked activations from the normal images to become more prominent. In Table 4 we illustrate such recovery results: after using a  $3 \times 3$  average filter on identified adversarial examples, the classification accuracy improved from almost 0% to 73.0%, showcasing the effectiveness of this simple average filter.

Those results show that we can both detect and recover from adversarial examples with high accuracy. But the main reason we performed this (overly simplistic) experiment is to show how simple it might be to cancel out some adversarial perturbations. Importantly, this result indicates that current deep convolutional networks are too locally focused: these are corruptions that can be cancelled out by a simple  $3 \times 3$  average filter, however they can adversely impact the entire result of the deep network. For human with a large receptive field, they will not even care about what happens

Table 4. Recovery Results. Simply using a  $3 \times 3$  average filter we can recover a large proportion of adversarial examples after detecting them using the algorithm described previously. More complex cancellation approaches such as foveation in [16] that utilizes cropping can achieve better results.

Approach	Top-5 Accuracy (Recovered Images)
Original Image (Non-corrupted)	86.5%
$3 \times 3$ Average Filter	73.0%
$5 \times 5$ Average Filter	68.0%
Foveation (Object Crop MP) [16]	82.6%

within a  $3 \times 3$  area. Therefore, we believe that future deep learning approaches should focus on enlarging the receptive field in order to reduce the chance of being fooled by adversarial examples. Another potential direction is to research classification approaches that do not require a softmax-type normalization, in order to avoid regularizing attacks such as the ones used in the adversarial optimization in (2).

## 8. Conclusion

This paper proposes an approach that detects the adversarials using simple statistics on convolutional layer outputs. A cascade classifier was designed based on simple statistics on filter outputs from each layer. And it was capable of detecting more than 85% of the adversarial examples. Experiments showed that our cascade classifier significantly outperform state-of-the-art on detecting adversarial examples. Experiment also showed transfer learning capabilities of our classifier, since the classifier we trained with LBFGS adversarials are capable of detecting EA-adversarials as well. Insights drawn from these experiments lead us to perform simple  $3 \times 3$  average filter to corrupted images, which successfully recovered most of them. In the future, we would like to explore GAN-type generative adversarial networks from the current results, with multiple rounds of adversarial detection and counter-detection.



## References

- [1] A. Balsubramani. Learning to abstain from binary prediction. *arXiv preprint arXiv:1602.08151*, 2016.
- [2] A. Bendale and T. E. Boulton. Towards open set deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- [4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
- [5] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [6] S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.
- [7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2001.
- [8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [9] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.
- [10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [11] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.
- [12] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2-3):245–272, 2010.
- [13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [14] G. R. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2003.
- [15] L. Li, M. L. Littman, T. J. Walsh, and A. L. Strehl. Knows what it knows: a framework for self-aware learning. *Machine learning*, 82(3):399–443, 2011.
- [16] Y. Luo, X. Boix, G. Roig, T. A. Poggio, and Q. Zhao. Foveation-based mechanisms alleviate adversarial examples. *arXiv preprint arXiv:1511.06292v3*, 2016.
- [17] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015.
- [18] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- [19] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *International Conference on Computer Vision (ICCV)*, 2015.
- [20] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1511.04508*, 2015.
- [21] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [22] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet. Adversarial manipulation of deep representations. 2016.
- [23] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.
- [24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv:1409.4842*, 2014.
- [26] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- [27] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.
- [28] Y. Wiener and R. El-Yaniv. Agnostic selective classification. In *Advances in Neural Information Processing Systems*, pages 1665–1673, 2011.
- [29] M. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*, 2013.
- [30] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.